Engineering 303  Digital Logic Design

LAB 2: Combinatorial Logic Design

Build the following designs and fully document them in your report.

**Deliverables:**
0) A Simple Verilog Combinatorial Circuit
1) A Simple Block Diagram Combinatorial Circuit
2) 2 XOR Equivalent Circuits - Block Diagram Implementation
3) Testing for Equivalent Circuits

**Demonstration Requirement:**
Download the equivalent circuits from Part 3 to the DE2 board and demo it to the instructor.

**Part 0 – A Simple Combo Verilog Simulation**
This part is more practice entering and simulating Verilog combinatorial circuits. Enter and simulate the designs given below. Refer to Lab 1 for “Verilog HDL File” entry procedure using a project name of “SimpleCombo”. Note we are creating project names that represent our design and don’t conflict with our previous project names stored on our USB drive.

Simulate your design with a grid time to 100ns and End Time to 800ns (or 0.800us). Note for 3 input signals we want to test all $2^3$ or eight combinations. This will require 8 windows of 100ns or 800ns total simulation time.

*Helpful Hint: You may set several signals at once using the count feature.* To do this, highlight the three input signals, **right click to group them**, select a count to set them as an incrementing binary sequence (eg 000, 001, 010, 011, 100, 101, 110,111), then ungroup them. It should look like this now:
Use Excel or Word to **manually create the truth table** for these equations and include it in your report. Simulate your design and verify that the simulation waveform matches the truth table.

Include the **Verilog design entry** and the **simulator waveform** in your report. Use Snipping Tool to paste into your report. Select only the design entry or simulator waveform window.

It should look similar to the images shown above.
Part 1: A Simple Combo Block Diagram Simulation

Repeat the previous design except use a block diagram simulation, as shown below. These are the same equations from the previous part. Refer to Lab 1 for “Block Diagram/Schematic File” entry procedure using a project name of “SimpleComboDiagram”.

Here the input signals are shown kept as a group for convenience. Note how this waveform has a glitch in \( f_1 \), whereas the Verilog design does not. This happens sometimes! Glitches happen, but it is important that the period is long enough to allow the waveform to stabilize. That is why we use a 100ns period in this class. Latter in the course we will learn how to identify and eliminate glitches in our design.

In large real life designs, we would want to make the clock period as small as possible so that the circuit can be run faster. Although we must use the simulation tool to identify possible glitches that could cause problems.

Simulate your design and verify that the waveform matches the truth table. Include the block diagram and the waveform in your report.
Part 2 – XOR Equivalent Circuits -- Block Diagram Implementation

The following are two XOR equivalent circuits. You can use DeMorgan’s theorem to prove these are equivalent. For now we are going to construct both to demonstrate their equivalence.

Build, simulate and verify correct operation of BOTH these designs in Quartus. Use the project names “EquivXOR1Diagram” and “EquivXOR2Diagram”. Include both the block diagram and the waveform for each design in your report.

\[ F = X \text{ exclusive-or } Y \]

The first one should look like the figure below in Quartus.

Complete BOTH designs. Simulate your designs and verify that the waveform matches the XOR truth table. Include the XOR truth table, the circuit block diagrams and the waveforms in your report.
Part 3 – Testing For Equivalent Circuits in Verilog

Build 2 circuit functions using Verilog to test if the following equations are logically equivalent. If the circuits are equivalent, they should produce the same output. Manually construct the truth tables for both of these circuits and compare the waveform outputs with each other and with the truth tables to determine if these two circuits are equivalent. You could combine both equations into one design if you wish.

Suggested project name “EquivTest” for your project.

Use the following equations:

\[ f_1 = b'(a \oplus c) + ab \]
\[ f_2 = a(b + c') + a'b'c \]

Translating this into Verilog would be:

```verilog
assign f1 = (~b & (a^c)) | (a & b);
assign f2 = a & (b | ~c) | (~a & ~b & c);
```

Use the following pin assignments to demonstrate your design on the DE2 board

<table>
<thead>
<tr>
<th>LAB 2 SIGNAL</th>
<th>DE2 Name</th>
<th>DE2 Pin</th>
<th>DE2-115 Pin</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>SW[17]</td>
<td>PIN_V2</td>
<td>PIN_Y23</td>
</tr>
<tr>
<td>b</td>
<td>SW[16]</td>
<td>PIN_V1</td>
<td>PIN_Y24</td>
</tr>
<tr>
<td>c</td>
<td>SW[15]</td>
<td>PIN_U4</td>
<td>PIN_AA22</td>
</tr>
<tr>
<td>f1</td>
<td>LEDR[1]</td>
<td>PIN_AF23</td>
<td>PIN_F19</td>
</tr>
<tr>
<td>f2</td>
<td>LEDR[0]</td>
<td>PIN_AE23</td>
<td>PIN_G19</td>
</tr>
</tbody>
</table>

Document in your report the truth table, Verilog design, waveforms, and conclusion if they are equivalent.